สร้างภาพเคลื่อนไหวและกราฟิกสามมิติโดยใช้ OOP

ศึกษาวิธีการเขียนโปรแกรมโดยใช้โอโอพีในเทอร์โบปาสคาล

ถ้าคุณเคยได้ยินคำว่า "การเก็บของเก่ามาใช้ใหม่" ในทางการเขียนโปรแกรมคุณคงนึกถึงการนำโปรแกรมเก่า ๆ ที่ได้เขียนนำมาพัฒนา เพื่อเอามาใช้ใหม่ ซึ่งทำให้ต้องเสียทั้งเวลาและกำลังความคิดมากทีเดียว ดังนั้นจึงมีคนพยายามผลิตสิ่งที่จะตอบสนองแนวคิดของการนำของเก่ากลับมาใช้ใหม่ ซึ่งถ้าคุณเคยรู้จักกับคำว่า OOP (Object Oriented Programming) คุณคงจะรู้ว่าการนำของเก่ากลับมาใช้นั้นไม่ใช่เรื่องยากอีกต่อไป เพราะโปรแกรมที่เขียนตามแนวทางโอโอพีนั้นจะมีโครงสร้างที่ดีกว่า เป็นระเบียบมากกว่าการเขียนโปรแกรมแบบเดิม และสามารถนำของเก่ากลับมาใช้และพัฒนาต่อไปได้สะดวก รวดเร็ว

ในบทความนี้จะนำเสนอการสร้างภาพเคลื่อนไหวและภาพกราฟิกสามมิติโดยใช้การเขียนโปรแกรมภาษาปาสคาลในแบบโอโอพี เนื่องจากในภาษาปาสคาลนั้นถึงแม้จะมีฟังก์ชันทางด้านกราฟิกให้ใช้ได้มากมาย แต่การสร้างภาพบางอย่างยังต้องอาศัยการเขียนโปรแกรมซึ่งค่อนข้างยุ่งยากซับซ้อน เช่น การทำภาพเคลื่อนไหวและการสร้างภาพสามมิติ โปรแกรมที่เขียนด้วยแนวเดิมนั้นเมื่อนำไปต่อรวมเข้ากับโมดูลอื่น ๆ มักเกิดปัญหาขึ้น เช่น ต้องแก้ไขตัวแปร ดังนั้นจุดประสงค์หลักของบทความนี้ คือนำเสนอโปรแกรมสร้างภาพกราฟิกที่สามารถนำไปประกอบกับโมดูลอื่น ๆ หรือนำไปพัฒนาต่อไป ทำให้เกิดการเรียนรู้กระบวนการเขียนโปรแกรมในแบบโอโอพีด้วย

เริ่มต้นกำหนดออปเจ็ค (Object)

โอโอพีมีหลักการพื้นฐานอยู่ 5 อย่าง คือ Object, Message, Method, Class, Inheritance ซึ่งออปเจ็คก็คือโค้ด (code) และข้อมูล (data) ในส่วนอื่น ๆ จะอธิบายต่อ ๆ ไปในเนื้อหา

ก่อนทำการเขียนโปรแกรมแบบโอโอพีจะต้องทำการกำหนดออปเจ็คเทียบได้กับเรคอร์ด (record) ในภาษาปาสคาลก่อน โดยกำหนดลงใน Type รายละเอียดต่าง ๆ ของวัตถุ ที่จะนำมาทำเป็นภาพเคลื่อนไหวถูกเก็บอยู่ในยูนิต Test01 ดังรูปที่ 1

Location กำหนดเป็นออปเจ็คประกอบด้วยตำแหน่ง X, Y ที่จะนำวัตถุไปวางไว้ตอนเริ่มต้นและกำหนด method Constructor Init ไว้ในขณะนิยามออปเจ็คนี้ด้วย ในทางปฏิบัติจะไม่เข้าถึงตัวแปรในออปเจ็คโดยตรง แต่จะกระทำผ่านโพรซีเดอร์หรือฟังก์ชัน ซึ่งต่อไปจะเรียกว่า method (method หรือ วิธีการ คือโค้ดที่ประกอบกันเป็นออปเจ็คสามารถจัดการกับ message คืออาร์กิวเมนต์และตัวเลือกต่าง ๆ ได้) ในบทความได้กล่าวถึง Construtor แต่ไม่ได้ใช้ procedure เพราะต้องการบอกให้คอมพายเลอร์รู้ว่า method นี้เป็นตัวแรกที่จะเรียกใช้ตัวแปรออปเจ็ค เพื่อสร้าง VMT (Virtual Method Table) ของออปเจ็คขึ้นมา ซึ่งจำเป็นสำหรับออปเจ็คที่ใช้คำสงวน virtual เพราะโปรแกรมจะได้ไปดูค่าแอดเดรสที่จะกระโดดไปทำงานต่อจาก VMT นี่เอง การใส่ virtual ตามหลังจากชื่อ method จะใส่ในกรณีที่ method มีชื่อซ้ำกันแล้วต้องการให้การทำงานของ method ทำงานตามสิ่งที่อยู่ภายในออปเจ็คของมันเองเท่านั้น

```
    Unit Test01;
Interface
Uses Crt,Graph;
Type
   Location = Object
       X,Y : Integer;
       Visible : Boolean;
       constructor Init(InitX,InitY : Integer);
   end;

   DestPoint = object(Location)
       constructor Init(InitX,InitY : Integer);
       procedure Show;
       procedure Hide;
       procedure MoveTo(XX,YY : Integer);  virtual;
   end;

Implementation

Const
   Image : array[1..64] of byte = ( $27 , $00 ,$0b ,$00
                    ,$00 , $00 ,$1e ,$00 ,$00
                    ,$00 , $00 ,$1f ,$00 ,$00
                    ,$00 , $38 ,$1f ,$80 ,$00
                    ,$40 , $3e ,$1f ,$c0 ,$00
                    ,$98 , $ff ,$ff ,$ff ,$e0
                    ,$a6 , $ff ,$ff ,$f0 ,$fe
                    ,$94 , $ff ,$ff ,$ff ,$e0
                    ,$48 , $3e ,$1f ,$c0 ,$00
                    ,$a0 , $3c ,$1f ,$80 ,$00
                    ,$00 , $00 ,$1f ,$00 ,$00
                    ,$00 , $00 ,$1e ,$00 ,$00
                    ,$00 , $00 ,$00 ,$00 ,$00);

constructor Location.Init(InitX,InitY : Integer);
begin
       X := InitX;
       Y := InitY;
       Visible := False;
end;

constructor DestPoint.Init(InitX,InitY : Integer);
begin
       Location.Init(InitX,InitY);
       Visible := False;
end;

procedure DestPoint.Show;
begin
       Visible := True;
       PutImage(X,Y,Image,0);
end;

procedure DestPoint.Hide;
begin
       Visible := False;
end;

procedure DestPoint.MoveTo(XX,YY : Integer);
begin
     If Visible Then Hide;
     X := XX;
     Y := YY;
     Show;
end;
end.
```

รูปที่ 1 ยูนิต Test01.pas

ซึ่งช่วยลดข้อผิดพลาดของการไปเรียกสิ่งที่อยู่ในออปเจ็คอื่นมาใช้

สืบต่อลูกหลาน (Inheritance)

ออปเจ็คถัดมา คือออปเจ็ค Distpoint ซึ่งเป็นออปเจ็คที่สืบต่อลงมาจากออปเจ็ค Location โดยเขียนเป็น Distpoint = object (Location) เพื่อแสดงว่าออปเจ็ค Distpoint มีความสามารถใช้ทุกสิ่งที่อยู่ในออปเจ็ค Locationได้ ถัดลงมาเป็นการให้ค่าของข้อมูลซึ่งแทนภาพเครื่องบินเจ็ต ที่จะนำมาทำเป็นภาพเคลื่อนไหว ในส่วนนี้นำไปดัดแปลงเป็นข้อมูลอื่น ๆ ได้ ส่วนโปรแกรมของ method จะกระทำอยู่นอกออปเจ็ค ซึ่งจะขึ้นต้นด้วยชื่อออปเจ็คแล้วตามด้วยจุด จากนั้นจึงใส่ชื่อของ method ใน method Distpoint.MoveTo กำหนดให้รับค่า NewX,NewY ซึ่งเป็นตำแหน่งสุดท้ายที่ภาพจะเคลื่อนที่

สร้างภาพเคลื่อนไหว

การเรียกใช้ยูนิตที่กล่าวมาแล้วอยู่ในโปรแกรม OOPMOVE ในรูปที่ 2

โปรแกรมเริ่มจากการเรียกใช้งาน Jet.Init เพื่อให้ค่าพิกัดเริ่มต้น แล้วแสดงรูปภาพ ณ.ตำแหน่งนั้นด้วย Jet.Show กำหนดให้เครื่องบินเคลื่อนที่ไปตามแนวแกน x ทีละ 5 จุด โดย Jet.MoveTo ทำงานอยู่ในลูปพร้อมกับมีเสียงประกอบ ในบทนี้ไม่ได้ลบภาพเก่าออกเมื่อมีการเคลื่อนที่ เพราะต้องการให้เกิดความสวยงามของภาพเครื่องบินที่บินต่อเนื่องไป

กราฟิกสามมิติ

สำหรับการสร้างภาพกราฟิกสามมิตินี้จะอธิบายถึงรายละเอียดการคำนวณค่ามุม การหมุน และมุมมองต่าง ๆ ประกอบไปด้วย เพื่อจะได้ทำความเข้าใจโปรแกรมได้ง่ายขึ้น การสร้างภาพสามมิตินี้ มีวิธีกระทำได้สองวิธี คือการใช้เวคเตอร์ (vector) และการลงบิตแม็พ ในที่นี้จะใช้วิธีการคำนวณหาจุดต่าง ๆ ของเวคเตอร์เพื่อจะนำมากำหนดลงในระนาบประกอบกันเป็นภาพสามมิติ

การเคลื่อนที่ของวัตถุสามมิติ

วัตถุสามมิติใด ๆ มีการเคลื่อนที่ได้สองทาง คือเคลื่อนที่เป็นทางตรงตามแนวแกน x,y,z และหมุนรอบแกน x,y,z โดยหมุนพิกัด A(x,y,z) ไปรอบ

```
Program Graphic_3D_By_OOP;
Uses  Crt,Graph,Test02;
const XYZ : array[1..4,1..4] of ThreeVal = (
  ((x1:-10;y1:8;z1:0),(x1:10;y1:8;z1:0),(x1:10;y1:0;z1: 8),(x1:-10;y1:0;z1: 8)),
  ((x1:-10;y1:8;z1:0),(x1:10;y1:8;z1:0),(x1:10;y1:0;z1:-8),(x1:-10;y1:0;z1:-8)),
  ((x1:-10;y1:0;z1:8),(x1:10;y1:0;z1:8),(x1:10;y1:0;z1:-8),(x1:-10;y1:0;z1:-8)),
  ((x1:-10;y1:0;z1:8),(x1:-10;y1:0;z1:-8),(x1:-10;y1:8;z1:0),(x1:-10;y1:0;z1:8)));
Var
     ThreeD               : ThreeDpoint;
     gdriver,gmode,i,j    : integer;
     poly                 : array[1..4,1..4] of TwoVal;

Procedure CheckKey;
Var  Ch : Char;
Begin
     Repeat Until Keypressed;
     Ch := Readkey;
     If Ch = #27 Then Halt(0);
End;

Begin { * M a i N * }
     gdriver := detect;
     initgraph(gdriver,gmode,' ');
     ThreeD.Init(30.0,45.0,30.0);
     ThreeD.Rotate;
     For i := 1 to 4 do
     begin
        For j := 1 to 4 do
        begin
           ThreeD.Plots(XYZ[i,j],XYZ[i,j]);
           ThreeD.Sline(XYZ[i,j],poly[i,j]);
        end;
        drawpoly(4,poly[i]);
     end;
     CheckKey;
     closegraph;
end.
```

รูปที่ 2 โปรแกรม OOPMOVE

แกน x,y,z การคำนวณค่าพิกัดของการหมุนมีสูตรที่คำนวณได้ดังนี้

การหมุนรอบแกน x จะได้ค่า x เป็นค่าพิกัดที่ใส่เข้ามาคือ Ax ได้ค่า y จากการคำนวณ y = Ay*cose − Az*sine และค่า z = Ay*sine + Az*cose

การหมุนรอบแกน y จะได้ค่า y เป็น Ay ค่า x = Az*sine + Ax*cose ค่า z = Az*cose − Ax*sine

การหมุนรอบแกน z จะได้ค่า z เป็น Az ค่า x = Ax*cose − Az*sine ค่า y = Ax*sine + Ay*cose

การหาตำแหน่ง x,y ที่จะนำค่า x,y,z ที่ได้ไปพล็อตลงบนจอนั้น อาศัยหลักการมองระนาบของวัตถุขนานอยู่กับแกน xy โดยวัตถุมีพิกัดอยู่ที่ A(x,y,z) โดยมีจุด 0,0,0 และจุดกึ่งกลางของจออยู่จุดเดียวกัน ตาที่มองอยู่ในพิกัดมีระยะทางทางแกน z เป็นค่าระยะห่างระหว่างวัตถุกับระนาบ xy รวมกับระยะห่างระหว่างตากับระนาบ ซึ่งในที่นี้จะให้ตำแหน่งของตาคือ 0,0,P1+P2 เมื่อคำนวณตามหลักคณิตศาสตร์แล้วจะได้ค่าที่อยู่บนระนาบเดียวกันกับวัตถุที่อยู่เป็น x = (P2/(P1 + P2 − Az))*Ax,y = (P2/(P1 + P2 − Az))*Ay เมื่อจะนำมาลงบนระนาบ xy ของจอภาพต้องคำนึงถึงความละเอียดของจอภาพด้วย โดยคำนวณค่าคงที่ทางแกน x,y ค่าหนึ่งเพื่อนำมาคูณเข้ากับค่า x,y ที่ได้ ถ้าจอภาพมีขนาด 26*20 cm. ค่าคงที่นี้คือ xpcm = getmaxx/26,ypcm = getmaxy/20 ดังนั้นค่าจุด x,y ที่จะนำมาพล็อตบนจอภาพได้จะคำนวณได้จาก

```
X = (P2/(P1 + P2 − Az))
    *Ax*xpcm + getmaxx div 2
Y = getmaxy div 2 − (P2/(P1 + P2 − Az))
    *Ay*ypcm
```

รูปที่ 3 เป็นยูนิตของการกำหนด

รูปที่ 3 เป็นยูนิตที่ใช้กำหนดตำแหน่งและคำนวณหาค่าพิกัดของด้านต่าง ๆ

```
    Unit Test02;
Interface
Uses Crt,Graph;
Type
      TwoDpoint  = object
              X1,Y1 : Real;
              procedure Init(InitX,InitY : Real);
      end;

      TwoVal = record
              x1:integer; y1 : integer;
      end;
      ThreeVal = record
             x1 : real; y1 : real; z1 : real;
      end;

      ThreeDpoint = object(TwoDpoint)
              Z1 : Real;
              procedure Init(InitX,InitY,InitDeep : Real);
              procedure Rotate;
              procedure Plots(Val1: ThreeVal;var Val2:ThreeVal);
              procedure Sline(Val1 : ThreeVal;var Val2 : TwoVal);
      end;

Implementation

var   sin_x,sin_y,sin_z,cos_x,cos_y,cos_z : Real;

procedure TwoDpoint.Init(InitX,InitY : Real);
begin
      X1 := InitX;
      Y1 := InitY;
end;

procedure ThreeDpoint.Init(InitX,InitY,InitDeep : Real);
begin
      TwoDpoint.Init(InitX,InitY);
      Z1 := InitDeep;
end;

procedure ThreeDpoint.Rotate;
begin
      sin_x := sin(X1*pi/180);
      sin_y := sin(Y1*pi/180);
      sin_z := sin(Z1*pi/180);
      cos_x := cos(X1*pi/180);
      cos_y := cos(Y1*pi/180);
      cos_z := cos(Z1*pi/180);
end;

procedure ThreeDpoint.Plots(Val1:ThreeVal;var Val2: ThreeVal);
var   P,Q,R : ThreeDpoint;
begin
      P.X1 := Val1.x1;
      P.Y1 := cos_x*Val1.y1-sin_x*Val1.z1;
      P.Z1 := sin_x*Val1.y1+cos_x*Val1.z1;
         Q.X1 := sin_y*P.Z1+cos_y*P.X1;
         Q.Y1 := P.Y1;
         Q.Z1 := cos_y*P.Z1-sin_y*P.X1;
      R.X1 := cos_z*Q.X1-sin_z*Q.Y1;
      R.Y1 := sin_z*Q.X1+cos_z*Q.Y1;
      R.Z1 := Q.Z1;
      Val2.x1 := R.X1;
      Val2.y1 := R.Y1;
      Val2.z1 := R.Z1;
end;
```

รูปที่ 3 (ต่อ)

```
procedure ThreeDpoint.Sline(Val1 : ThreeVal;var Val2 : TwoVal);
const  P1 = 100.0; P2 = 100.0;
var    xpcm,ypcm : real;
begin
       xpcm := getmaxx/26; ypcm := getmaxy/20;
       Val2.x1 := round(P2/(P1+P2-Val1.z1)*Val1.x1*xpcm)+(getmaxx div 2);
       Val2.y1 := (getmaxy div 2)-round(P2/(P1+P2-Val1.z1)*Val1.y1*ypcm);
end;
end.
```

```
Program Graphic_Movement_OOP;
Uses Crt,Graph,Test01;
Var
    gdriver,gmode,x : Integer;
    Jet             : DestPoint;

Procedure CheckKey;
Var  Ch : Char;
Begin
     Repeat Until Keypressed;
     Ch := Readkey;
     If Ch = #27 Then Halt(0);
End;

Begin  { * M a i N * }
     gdriver := Detect;
     InitGraph(gdriver,gmode,' ');
     settextstyle(0,0,3);
     outtextxy(130,70,'GRAPHIC - OOP TECHNIQUE');
     settextstyle(3,0,5);
     outtextxy(320,130,'Created By');
     outtextxy(250,150,'Miss.ATCHARA  JANTARASAENGARAM');
     outtextxy(270,170,'APPLIED COMPUTER SCIENCE');
     outtextxy(160,190,'King Mongkut''s Institute of Technology  North Bangkok');
     Jet.Init(0,50);
     Jet.Init(0,100);
     Jet.Show;
     x:=0;
     while x < 700 do begin
        x:=x+5;
        Jet.MoveTo(x,50);
        Jet.MoveTo(x,100);
        sound(50+x*2);
        delay(100);
     end;
     nosound;
     CheckKey;
     closegraph;
end.
```

รูปที่ 4 โปรแกรมสร้างภาพสามมิติ

ตำแหน่งและคำนวณหาค่าพิกัดของด้านต่าง ๆ ที่จะนำมาพล็อตเป็นภาพสามมิติต่อไป

จากรูปที่ 3 จะเห็นว่าออปเจ็ค ThreeDpoint มีการสืบต่อจากออปเจ็ค TwoDpoint ในออปเจ็ค ThreeDpoint จะ declare method Rotate ซึ่งใช้ในการคำนวณค่า sin,cos Plots จะคำนวณ

หาพิกัดที่จะนำไปพล็อตบนจอภาพและ Sline ใช้ในการเก็บพิกัดต่าง ๆ ไว้ การเรียกใช้ยูนิตนี้อยู่ในรูปที่ 4 โปรแกรมสร้างภาพสามมิติ โดยให้ค่าพิกัดต่าง ๆ ของรูปทรงสามเหลี่ยมไว้เป็นค่าคงที่

ค่ามุมที่ให้หมุนรอบแกน x,y,z ในที่นี้ให้เป็น 30,45,30 ตามลำดับ เมื่อคำนวณค่ามุม sin,cos จาก ThreeD.-Rotate ได้แล้ว นำมาเข้าลูปเพื่อใช้ ThreeD.Plots คำนวณหาจุดพิกัดต่าง ๆ ที่จะนำไปพล็อตบนจอภาพแล้วเก็บไว้ในอะเรย์เพื่อนำมาลากเส้นเชื่อมกันโดยฟังก์ชัน drawpoly ให้เกิดภาพสามมิติขึ้นมา

ถึงตรงนี้ ผู้อ่านคงพอมองเห็นแนวทางในการนำโอโอพีไปใช้ในการเขียนและพัฒนาโปรแกรมด้านกราฟิกและด้านอื่น ๆ ได้ ถ้าผู้อ่านจะนำโปรแกรมนี้ไปปรับปรุงเปลี่ยนแปลงต่อไปให้สมบูรณ์ขึ้นก็สามารถทำได้ ซึ่งอาจเพิ่มออปเจ็คต่าง ๆ เข้าไปอีกก็ได้ ท้ายสุดนี้หวังว่าบทความนี้คงจะให้ประโยชน์ได้บ้างไม่มากก็น้อย สวัสดีค่ะ